International Journal of Electrical and Computer Engineering (IJECE)
Vol. 14, No. 1, February 2024, pp. 819~829
ISSN: 2088-8708, DOI: 10.11591/ijece.v14i1.pp819-829

Generate fuzzy string-matching to build self attention on Indonesian medical-chatbot

Wiwin Suwarningsih, Nuryani
Research Center for Data and Information Sciences, National Research and Innovation Agency, Bandung, Indonesia

Article Info

Article history:

Received Jun 22, 2023
Revised Jul 25, 2023
Accepted Oct 20, 2023

Keywords:

Fuzzy string matching
Indonesian medical
Low resource language
Self-attention
Transformers

ABSTRACT

Chatbot is a form of interactive conversation that requires quick and precise answers. The process of identifying answers to users' questions involves string matching and handling incorrect spelling. Therefore, a system that can independently predict and correct letters is highly necessary. The approach used to address this issue is to enhance the fuzzy string-matching method by incorporating several features for self-attention. The combination of fuzzy string-matching methods employed includes Jaro Winkler distance + Levenshtein Damerau distance and Damerau Levenshtein + Rabin Carp. The reason for using this combination is their ability not only to match strings but also to correct word typing errors. This research contributes by developing a self-attention mechanism through a modified fuzzy string-matching model with enhanced word feature structures. The goal is to utilize this self-attention mechanism in constructing the Indonesian medical bidirectional encoder representations from transformers (IM-BERT). This will serve as a foundation for additional features to provide accurate answers in the Indonesian medical question and answer system, achieving an exact match of 85.7% and an F1-score of 87.6%.

Corresponding Author:

Wiwin Suwarningsih
Research Center for Data and Information Sciences, National Research and Innovation Agency
Tower 2 Floor 5, Sangkuriang street 21 Bandung, West Java, Indonesia
Email: wiwin.suwarningsih@brin.go.id

1. INTRODUCTION

The simulation of conversations and interactive communication between humans and mobile devices through text has become widely known to the public. This interactive system is commonly referred to as chatbot. Knowledge is embedded in the system to identify sentences and determine appropriate answers during the question-and-answer process. The chatbot utilizes various string-matching methods, such as Bruteforce [1], [2], Knuth Morris Pratt (KMP) [3], [4], Boyer Moore (BM) [5], [6], and regular expression [7] to perform word searches and obtain correct and valid answers.

However, a weakness of the string-matching method is its inability to tolerate typos because it relies on exact matching algorithms that require exact string matches [8], [9]. This means that the algorithm must match the string pattern exactly regardless of the words in the pattern. For example, when searching for the pattern "*saya sakit*" (meaning "I am sick") in the texts "*saya sakit tapi masih suka makan*" (meaning "I am sick but still like to eat") and "*saya orang sakit*" (meaning "I am a sick person"), the system will only identify the first text as containing the pattern "*saya sakit*" while ignoring the second text. The search feature in chatbot systems is common and essential for every application [10], [11]. This feature allows users to find information that may be challenging to locate because of the large amount of data in the knowledge base, which often contains similar word and sentence patterns [12], [13]. Hence, there is a need for an inexact method with a high level of error tolerance; one such method is fuzzy string matching [14]. This concept involves searching for identical or closely related strings in a container or dictionary. Widely used fuzzy string-matching methods include Levenshtein distance [15], [16], Damerau-Levenshtein [17], [18], hamming [19], [20], and fuzzy logic [21].

The process of searching for text in documents by comparing two strings using a Levenshtein distance matrix is highly supportive of obtaining string matches [18]. This is achieved by analyzing each character defined in the matrix and determining the maximum value as the ratio of text similarity [15]. Similar letters were then grouped, and the weight difference between the groups was reduced [22]. Improving accuracy involves calculating the hash distance between texts being compared [23]. However, the string-matching system based on Levenshtein distance still lacks the ability to correct word errors. The Damerau-Levenshtein distance [18] is a method that addresses this issue. The Damerau-Levenshtein distance is capable of handling complex spelling variations [24], correcting spelling errors in password phrases that involve symbol and number combinations [19], and identifying a sequence of correction operations between two adjacent strings [20]. By reducing unnecessary string comparisons and avoiding duplicate outputs, accuracy can be improved [25], enabling quick prediction of letter corrections. Letter prediction and correction were supported using the inference rules in the fuzzy method. This method predicts words that fail to be recognized in speech recognition [21] and conversation systems [22]. Applying the fuzzy method using dynamic inference and modification of word feature structures presents an opportunity to enhance the reliability of natural language processing (NLP) in chatbot systems. In this study, we propose the construction of self-attention by generating a fuzzy string-matching model with modified word feature structures.

The contributions of this paper are as follows: i) building self-attention based on fuzzy string matching, ii) establishing Indonesian medical bidirectional encoder representations from transformers (IM-BERT), and iii) constructing an Indonesian medical dataset, which is a low-resource language. The uniqueness of the Indonesian language includes the usage of multiple languages within a single interrogative sentence, such as Indonesian combined with English or regional languages like Javanese, Sundanese, and Betawi. Additionally, Indonesian exhibits the presence of out-of-vocabulary language elements, including slang, word abbreviations, and Indonesianized English.

The main contents of this study are as follows. Section 2 describes the method, which includes the acquisition of data, preprocessing, comparison of the combination of fuzzy string matching, building self-attention, and an overview of IM-BERT. Section 3 presents the results and discussion and covers experimental results from multiple perspectives. Section 4 summarizes the research work of this study and clarifies directions for future research.

2. METHOD

We build a framework as shown in Figure 1 to implement the self-attention generation method using fuzzy string matching for dynamic inference in the IM-BERT model. First, online news was collected by crawling Indonesian online news portals and stored in a database. Preprocessing, conversion, and classification were then used to compare the fuzzy string-matching methods. In the final stage, a combined fuzzy string model was used to generate the self-attention model, which was stored in the database to form the IM-BERT model. A comprehensive explanation of each procedure is provided below.

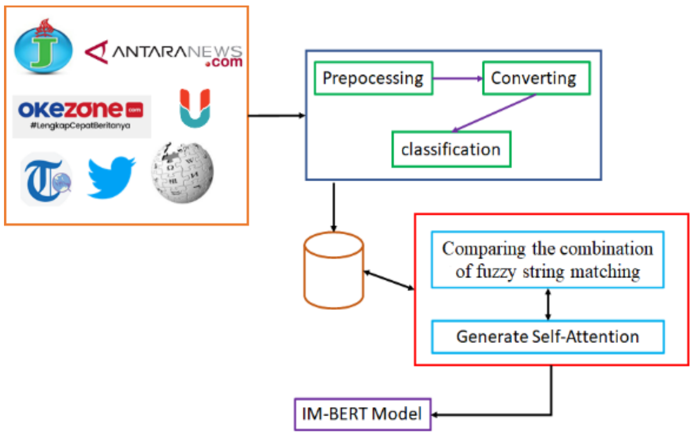


Figure 1. The system framework

2.1. Data acquisition

The input data for our study consisted of articles about coronavirus disease 2019 (COVID-19) news obtained through crawling results from various sources such as Indonesian twitter, Indonesian Wikipedia, Jakarta News (https://www.thejakartapost.com), Okezone (https://news.okezone.com), Antara (https://www.antaranews.com), Kumparan (https://kumparan.com), Tribune (https://www.tribunnews.com). These articles were processed and converted into an appropriate format for use as input data for our knowledge-based system. The dataset comprised 99,315 sentences, which were divided into 70% training data and 30% testing data. The dataset contained various sentence types. Various basic words, prefixed words, suffixed words, affixed words, conjunctions, pre-positions, and Indonesian standard words. Phrases consisting of multiple languages (e.g., "good *pagi*" meaning "good morning", "congrat *ya*" meaning "congratulations"). Out-of-vocabulary (OOV) words such as slang or "*alay*" (e.g., "*curhat*" meaning "a shoulder to cry on," "*woles*" meaning "relaxed," "*jutek*" meaning "bitchy," "*lebay*" meaning "excessive"), Indonesianized English words (e.g., "*ausam*" meaning "awesome," "*perigut*" meaning "very good," "*tengkyu*" meaning "thank you," "*polis*" meaning "police," "*donwori*" meaning "don't worry").

2.2. Data preprocessing

Text preprocessing, conversion, and classification are crucial steps for handling NLP tasks. Data preprocessing is conducted to address issues that may hinder data processing, primarily because of the inconsistent format of a significant portion of the collected crawled data. Given the diverse nature of data, an initial processing stage for standardizing language is highly necessary. For instance, Indonesianized English words such as "donwori" (meaning "don't worry") require conversion to overcome data inconsistencies. These steps involve converting the text into an appropriate format and categorizing it based on predetermined classes. After the conversion is completed, the subsequent step involves text classification, which entails grouping or categorizing the data according to predetermined rules, using Indonesian medical terms.

2.3. Comparing the combination of fuzzy string-matching method

To achieve optimal results, we compared different combinations of fuzzy methods: i) Levenshtein Damerau distance + Jaro Winkler distance and ii) Jaro Winkler distance + Rabin Carp. The purpose of this comparison was to determine the combination that yielded the highest percentage and add complexity to improve the results. The combination with the highest result was then used for dynamic inference to support the application of the self-attention method, similar to the bidirectional encoder representations from transformer (BERT) language model. This approach aims to support the development of IM-BERT by modifying the standard fuzzy method and incorporating self-attention features. The fuzzy comparison method involves several processes: i) Checking the Indonesian sentence patterns, ii) checking bulk documents, iii) continuous development of the dictionary to update and expand existing vocabularies, and iv) automatic replacement of word suggestions to immediately correct misspelled words.

2.3.1. Levensten Damerau distance versus Jaro Winkler distance

The Damerau-Levenshtein distance function operates on finite strings from the alphabet and returns an integer. The distance matrix is calculated based on strings S_1, S_2, and S_3, and it satisfies certain conditions: i) non-negativity: $d(S_1, S_2) \geq 0$; ii) non-degeneracy: $d(S_1, S_2) = 0$ if and only if $S_1 = S_2$; iii) symmetry: $d(S_1, S_2) = d(S_2, S_1)$; iv) triangle inequality: $d(S_1, S_2) + d(S_2, S_3) \geq d(S_1, S_3)$. The sorting process in the Damerau-Levenshtein distance involves determining the smallest value for each error. On the other hand, the Jaro-Winkler distance algorithm utilizes a different sorting process compared to the Damerau-Levenshtein distance. In the Jaro-Winkler distance algorithm, words are considered similar when the distance value is one, whereas in the other three algorithms, words are considered similar when the distance value is zero.

We combined the Damerau-Levenshtein distance and Jaro-Winkler distance algorithms as shown in Figure 2 to detect spelling mismatches in Indonesian words and transform words, including OOV words and multilanguage phrases. When a mismatch is detected between the spelling of a document and the spelling in the Indonesian medical dictionary and OOV dictionary, the system displays correct word suggestions using the Jaro-Winkler distance algorithm. Based on the displayed word choices, the misspelled words in the script were corrected by selecting an appropriate word solution. If no more spelling errors were found, the document was considered valid.

2.3.2. Jaro Winkler distance versus Rabin-Karp

The Jaro-Winkler distance algorithm is commonly used to measure the similarity between two strings and is often employed to detect plagiarism cases in documents [26], [27]. This algorithm is faster and more efficient for short strings owing to its effective quadratic-runtime complexity. The Jaro-Winkler distance algorithm involves three steps: i) calculating the length of the strings [28], ii) determining the number of similar characters between the two strings [29], and iii) determining the number of transpositions [30]. The Jaro-Winkler distance (d_j) is calculated using (1) for two strings, s_1 and s_2:

$$d_j = \frac{1}{3} x \left(\frac{m}{s_1} + \frac{m}{s_2} + \frac{m-t}{m} \right) \quad (1)$$

where m is the number of similar characters, s_1 is the length of string-1, s_2 is the length of string-2, and t is the number of transpositions.

The Rabin-Karp algorithm utilizes the hash method for performing multiple searches [31]. The steps involved in Rabin-Karp include: i) removing punctuation marks from the document and converting the text to lowercase for the search; ii) dividing the texts into grams with a predefined k-gram value; iii) calculating the hash value using the rolling hash function for each gram, following the formula: $h=c_1*b_{k-1}|c_2*b_{k-2}|...|c_k-1*b|c_k$; iv) identifying matching hash values between two texts; and v) determining the similarity between two pieces of text using Dice's similarity coefficient equation.

$$S = \frac{2*c}{A+B} \quad (2)$$

where S represents the similarity value, C represents the same k-gram value of the text being compared, A represents the number of k-grams in the first text, and B represents the number of k-grams in the second text. We combined the two Jaro-Winkler distance algorithms and the Rabin-Karp algorithm as shown in Figure 3 by dividing the tasks into text inputs. The single-pattern text input was processed using the Jaro-Winkler distance. Multiple-pattern text inputs and OOV text were processed using the Rabin-Karp algorithm.

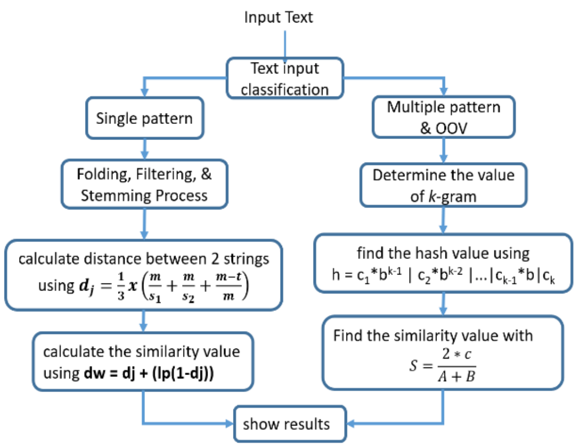

Input Text
Text input classification
Single pattern
Multiple pattern & OOV
Folding, Filtering, & Stemming Process
Determine the value of k-gram
calculate distance between 2 strings using $d_j = \frac{1}{3}x\left(\frac{m}{s_1}+\frac{m}{s_2}+\frac{m-t}{m}\right)$
find the hash value using $h = c_1*b^{k-1} | c_2*b^{k-2} |...|c_{k-1}*b|c_k$
calculate the similarity value using dw = dj + (lp(1-dj))
Find the similarity value with $S = \frac{2*c}{A+B}$
show results

Figure 2. Combination of Damerau Levenshtein distance with Jaro-Wingkler

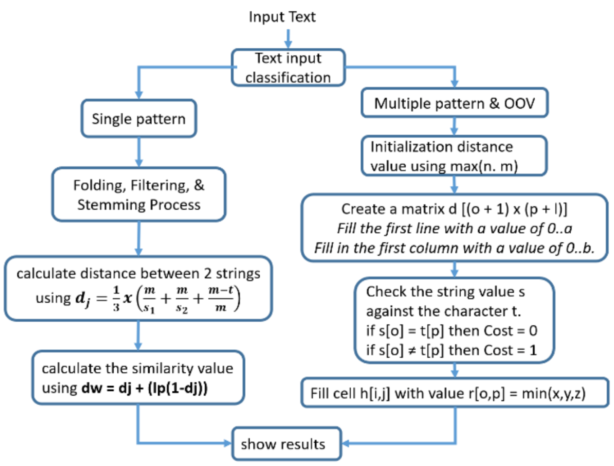

Input Text
Text input classification
Single pattern
Multiple pattern & OOV
Initialization distance value using max(n. m)
Folding, Filtering, & Stemming Process
Create a matrix d [(o + 1) x (p + l)]
Fill the first line with a value of 0..a
Fill in the first column with a value of 0..b.
calculate distance between 2 strings using $d_j = \frac{1}{3}x\left(\frac{m}{s_1}+\frac{m}{s_2}+\frac{m-t}{m}\right)$
Check the string value s against the character t.
if s[o] = t[p] then Cost = 0
if s[o] ≠ t[p] then Cost = 1
calculate the similarity value using dw = dj + (lp(1-dj))
Fill cell h[i,j] with value r[o,p] = min(x,y,z)
show results

Figure 3. Combination of Rabin-Karp with Jaro-Wingkler

2.4. Building self attention by generating fuzzy string-matching models

The combination of fuzzy string-matching methods aims to perform text classification while accounting for additional noise in the form of spelling errors [27]. The input data generated from the text classification results are used for the self-attention process [21]. The stages for building self-attention have been adapted from previous studies. The interaction stages between the inputs are illustrated in Figure 4.

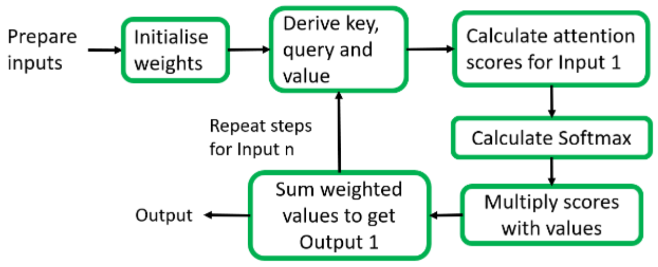

Prepare inputs
Initialise weights
Derive key, query and value
Calculate attention scores for Input 1
Calculate Softmax
Multiply scores with values
Sum weighted values to get Output 1
Repeat steps for Input n
Output

Figure 4. Interaction process between n inputs

As shown in Figure 4, several n inputs were generated from the selection results using a combination of fuzzy string-matching methods. The following stages involve the interactions between these inputs. The initial stage involves preparing the inputs by determining n inputs and m dimensions, resulting in $n \times m$ dimensions. The second step was the initialization of the input weights as the key, query, and value. In this study, we used a neural network to adjust the weights. The weights were assigned small numbers and then initialized using a random Gaussian distribution. This initialization was performed only once before the training.

The third step involves calculating the attention scores for Input-1 using the key representation for Input-1. We then applied the same set of weights to obtain the key representations for Input-2 and Input-3 and performed the same to obtain the value representation for each input. To accelerate this process, we add a bias vector to the matrix product. In the fourth step, we calculated the attention score for Input-1 by taking the dot product between the queries of Input-1 and all keys, including itself. With n key representations, we obtained attention scores. It is worth noting that we used only queries from Input-1. This step is known as dot product attention and is one of the several scoring functions. Other scoring functions include scale-point products and additives/concats.

The fifth step involves calculating the softmax function, which acts as the last activation function of the neural network. It normalizes the network output into a probability distribution based on the predicted output classes [32]. The standard softmax function is defined within the range of K [0, 1] and can be expressed using (3).

$$\sigma(z)_i = \frac{e^{z_i}}{\sum_{j=1}^{k} e^{z_j}} \; for \; i = 1, \dots, K \; and \; z = (z_1, \dots z_K) \in R^K \tag{3}$$

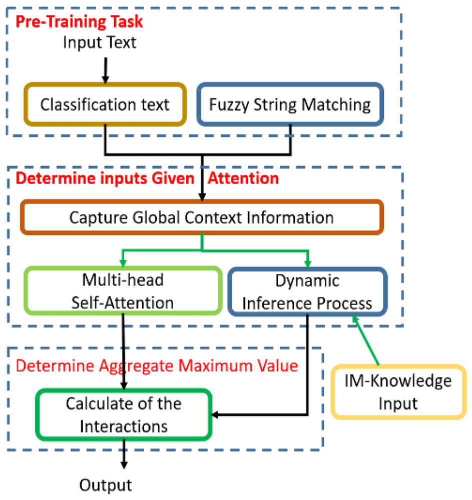
The sixth step involved multiplying the scores with the values. We determined the maximum attention score for each input and multiplied it by the corresponding value, resulting in n alignment vectors, known as weighted values. The seventh step involved adding the weighted values to obtain Output-1. We took all weighted values and added them element-wise, resulting in a vector [x, y, z] as Output-1. This was based on the query representation of Input-1 interacting with all the other keys, including itself. The last step was to repeat steps 4–7 for Input-2 and Input-3, producing Output-2 and Output-3, respectively.

2.5. Model overview IM-BERT

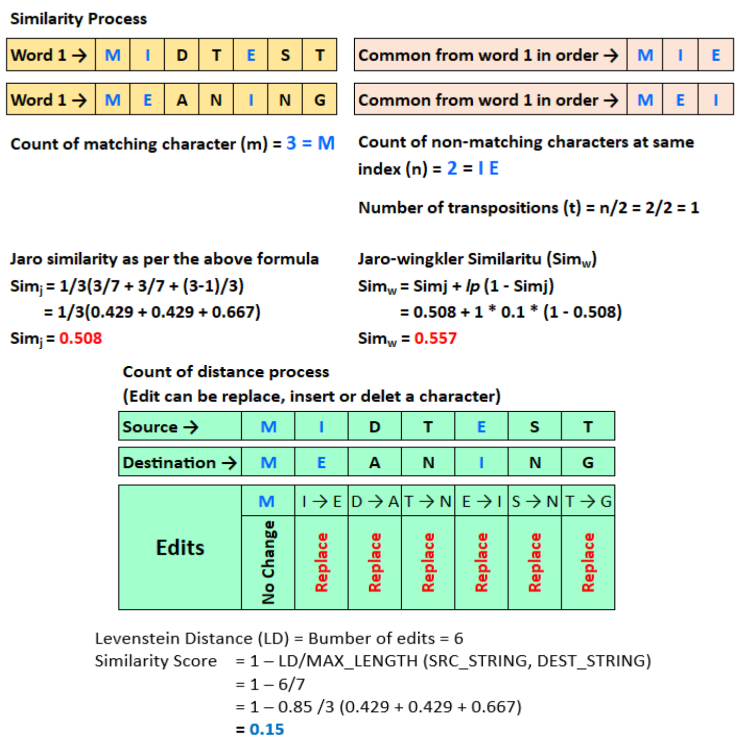
The self-attention mechanism used in this study, as depicted in Figure 4, was employed to capture global context information, and handle dynamic inference processes when capturing distant word features stored in the dictionary. The self-attention mechanism focuses on the target word and calculates the weight of dependence within a sentence. This approach helped alleviate distance-related issues between words in the sentence.

The main architecture of the proposed model is shown in Figure 5. In Figure 5(a), the model process of IM-BERT consists of three components: i) the text classification process, which handles additional noise in the form of spelling errors and OOV words. ii) The self-attention module allows inputs to interact with each other (“self”) and determine which inputs require more attention (“attention”). The output represents the aggregate of these interactions and scores generated from the *n*-input process. iii) Once the interaction process between inputs is complete, it is necessary to determine which inputs receive the most attention. This was achieved by calculating the maximum aggregate value of the interaction and attention scores.

In Figure 5(b) explain the simulation process for the pre-training task on input data, focusing on the comparison of two words, "midtest" and "meaning." This stage serves to illustrate the process of fuzzy string matching and distance calculation using the combination of models we employ. In this illustration, Figure 5(b) demonstrates the simulation process using Damerau-Levenshtein + Jaro-Winkler distance for similarity text process and count of distance process. The next step is determining inputs; attention involves considering input values, keys, queries, and outcomes, all of which are represented as vectors. The model proceeds by calculating the output as a weighted sum of these input values. Each value's weight is determined by a function that is compatible with the initial query and the corresponding key value.


Pre-Training Task
Input Text
Classification text
Fuzzy String Matching
Determine inputs Given Attention
Capture Global Context Information
Multi-head Self-Attention
Dynamic Inference Process
Determine Aggregate Maximum Value
Calculate of the Interactions
IM-Knowledge Input
Output

(a)



Similarity Process

Word 1 →	M	I	D	T	E	S	T
Word 1 →	M	E	A	N	I	N	G

Common from word 1 in order →	M	I	E
Common from word 1 in order →	M	E	I

Count of matching character (m) = 3 = M

Count of non-matching characters at same index (n) = 2 = I E

Number of transpositions (t) = n/2 = 2/2 = 1

Jaro similarity as per the above formula

$Sim_j = 1/3(3/7 + 3/7 + (3-1)/3)$

$= 1/3(0.429 + 0.429 + 0.667)$

$Sim_j = 0.508$

Jaro-wingkler Similaritu (Sim_w)

$Sim_w = Simj + lp\ (1 - Simj)$

$= 0.508 + 1 * 0.1 * (1 - 0.508)$

$Sim_w = 0.557$

Count of distance process
(Edit can be replace, insert or delet a character)

Source →	M	I	D	T	E	S	T
Destination →	M	E	A	N	I	N	G
Edits	M	I → E	D → A	T → N	E → I	S → N	T → G
	No Change	Replace	Replace	Replace	Replace	Replace	Replace

Levenstein Distance (LD) = Bumber of edits = 6
Similarity Score = 1 – LD/MAX_LENGTH (SRC_STRING, DEST_STRING)
= 1 – 6/7
= 1 – 0.85 /3 (0.429 + 0.429 + 0.667)
= 0.15

(b)

Figure 5. Model architecture of IM-BERT (a) model process of IM BERT and (b) simulation process using IM-BERT model

3. RESULTS AND DISCUSSION

3.1. The results of the comparison of the combination of fuzzy string-matching methods to find answers

This section presents the results of training and testing data using a combination of fuzzy string methods. This stage was performed to determine the best combination for matching words. The discussion consists of two combinations: i) the combination of the Levenshtein–Damerau distance + the Jaro-Winkler distance, and ii) the Rabin-Karp distance + the Jaro-Winkler distance.

3.1.1. Combination of Damerau Levenshtein distance with Jaro-Winkler distance (DL-JW)

The combination of the Damerau-Levenshtein distance and Jaro-Winkler distance involved dividing the functions as follows: The Damerau-Levenshtein distance performed word comparison by focusing on five types of errors: i) insertion of a letter or character, for example, “compny” was inserted 1 letter into “company”; ii) Deletion of a letter or character, for instance, the word “companye” was deleted by “company”; iii) Substitution of a letter with another letter, for example the word “compani” was replaced with “company”; iv) the sequential swapping of letters, for instance “cumpany” by changing the letters to “company” by changing the letters in this study, we added one type of error handling, that is; v) replacing words included in OOV with standard words.

The results of the Damerau-Levenshtein distance process were then compared with the spelling of each token using the Jaro-Winkler distance algorithm. The principle is to compare the query token with the list of words in the database. If there was a match, the processed word was considered correct, and the proximity distance was not calculated. If there was a discrepancy, the processed word was considered incorrect and the distance between the words in the database was calculated. A larger proximity value indicated a higher similarity between the compared words, whereas a smaller proximity value indicated less similarity. Out-of-vocabulary words were converted into standard words using a special OOV word dictionary.

The optimal use of these two algorithms helped detect inconsistencies in the spelling of Indonesian words (in Twitter texts or documents) compared to the data sources in the dictionary. The Damerau-Levenshtein distance algorithm corrects incorrect words and provides word suggestions based on the closest (smallest) distance between the spelling of the document and the spelling in the Indonesian dictionary. On the other hand, the Jaro-Winkler distance algorithm provides the correct word solution based on the high distance value obtained by comparing the word's spelling in the document with the word in the Indonesian dictionary.

3.1.2. Combination of Rabin Carp with Jaro-Winkler distance (RC-JW)

The second combination involves dividing the functions as follows. The Jaro-Winkler distance measures the similarity between two strings and detected sentences estimated to be the same as other sentences in the data dictionary. The higher the Jaro-Winkler distance between the two strings, the more similar they were. On the other hand, the Rabin-Karp algorithm was used to search for multiple patterns and out-of-vocabulary words in a combination of multilanguage sentences (Indonesian + regional language or Indonesian + English).

The Jaro-Winkler distance algorithm is effective in approximate string matching and produces accurate results for matching two short strings. This combination aims to overcome the limitations of the Rabin-Karp algorithm in the single-pattern search process by combining it with the Jaro-Winkler algorithm. The merging process involved preprocessing and forming K-gram data, followed by calculating the hash value of each K-gram using the Rabin-Karp algorithm. Subsequently, the Jaro-Winkler algorithm is used to find the same hash value and calculate the percentage level of similarity.

3.2. String matching training and testing results

In the experiments conducted on the combination of fuzzy string-matching algorithms, the aim was to evaluate the performance of the combination during both training and testing phases. The training data consisted of 76,213 sentences, whereas the testing data comprised 23,102 sentences. The training process utilized appropriate hardware and software to generate the optimal models.

The dataset used for training and testing contained a variety of sentences, including basic words, prefixed words, suffixed words, affixed words, conjunctions, prepositions, Indonesian standard words, phrases consisting of multiple languages, and out-of-vocabulary words. The training and testing of this combination of fuzzy string-matching algorithms involved the detection and correction of inappropriate Indonesian vocabularies. Corrections were made by providing correct word solutions, which involved processes such as replacement, exchange, insertion, deletion of characters, and transformation of OOV words from incorrect terms to correct ones.

Based on the data presented in Table 1, it can be concluded that the combination of the Rabin-Carp + Jaro-Winkler distance algorithm outperformed the combination of the Damerau Levenshtein distance + Jaro-Winkler distance during both training and testing. Further analysis revealed that the dataset used in the study exhibited a wide variety of sentence patterns, with a higher number of single patterns than multiple patterns. The Rabin-Carp + Jaro-Winkler distance algorithm showed a superior ability to transform words containing out-of-vocabulary (OOV) terms, as it required fewer steps in detecting and correcting incorrect and inappropriate words. The algorithm's use of k-Gram and determination of word weights facilitated the relevance judgment process, ensuring that the suggested words generated by the algorithm were appropriate. Additionally, the prefix scale value and length of the prefix provided indicators of character similarities and the number of similarities between the compared sentences.

Table 1. The results of string matching with a combination of fuzzy methods

Combination Method	Training Result					Testing Result				
	Delete	Add	Change	Swap	Transform	Delete	Add	Change	Swap	Transform
DL-JW Distance	0.74	0.89	0.70	0.71	0.72	0.80	0.92	0.80	0.79	0.75
RC-JW Distance	0.79	0.90	0.77	0.78	0.76	0.83	0.95	0.82	0.82	0.80

3.2.1. Determining the answers using self attention

The self-attention mechanism we built was used to determine the answers to Indonesian Medical question answering by weighting the possessed knowledge. In designing the question-answering training algorithm, the Rabin-Carp + Jaro-Winkler distance algorithm was employed to facilitate implementation. The knowledge weighting process consisted of the following steps: i) applying the Rabin-Carp + Jaro-Winkler distance ranking algorithm, ii) preprocessing the knowledge input, iii) initializing the knowledge using the Rabin-Carp + Jaro-Winkler distance algorithm, and iv) selecting the knowledge with the highest result.

After obtaining the knowledge with the highest weight, an assessment was performed using the IM-BERT question-answering model. The answering scoring process involved: i) assigning scores to the answers; ii) providing the IM-BERT question answering model with the input knowledge obtained through crawling; iii) initializing the IM-BERT question-answering (QA) model pipeline and tokenizer; and iv) processing the knowledge in the model and adding the candidate answers to an array. The answer collection was based on the highest score obtained from the IM-BERT question-answering model.

The answer-taking process follows these stages: i) ranking the candidate answers, ii) inputting the candidate answer array, and iii) sorting the candidate answers based on the highest score. Once the answers were obtained, it was necessary to evaluate the IM-BERT question-answering model. The evaluation utilized the k-fold algorithm with a statistical approach, including the following steps: i) providing input to the IM-BERT QA model using SQuAD data version 1.1, ii) filtering the SQuAD data based on question types, iii) evaluating the questions based on their types using k-fold, and iv) putting the evaluation results in JSON format.

3.2.2. Training the IM-BERT model

Once IM-BERT was pretrained with the available dataset, the training process was initiated. Prior to the training, the dataset was divided into three categories: training, validation, and testing. Owing to the large size of the datasets, loading them all into memory at once can overwhelm the system's memory capacity and slow down the program. To overcome this, a data loader is employed to efficiently manage the memory and enhance the training speed. The data loader, a function in PyTorch, acts as an iterator that combines the dataset and samples. Because the dataset was divided into three parts, three data loaders were utilized: one for training, one for validation, and one for testing.

During the training stage, BERT was fine-tuned by adjusting its hyperparameters. Most of the hyperparameters used for fine-tuning BERT remained the same as those used in regular BERT training. The hyperparameters employed in BERT training included: i) Batch size, which determines the number of samples entered the network before weight adjustment; and ii) Epoch, which represents the number of times the network processes the entire dataset. An epoch is completed when all instances have passed through the network during both forward and backward passes; iii) The learning rate, which determines the magnitude of weight changes in the neural network. A higher learning rate facilitates faster gradient movement towards optimal values.

3.2.3. Baseline

As a baseline for our proposed BERT architecture, we compared it with other architectures such as robustly optimized BERT approach (RoBERTa) and a lite BERT (ALBERT) using our existing dataset. The training process for Indonesian language models tends to be time-consuming because of limited resources. Therefore, we conducted a comprehensive training (referred to as "Long Training") for RoBERTa and ALBERT. The loss results during training showed a "trough" caused by changes in the corpus distribution.

This change occurred when transitioning from the Wikipedia corpus, which typically employs a standard language, to the Twitter corpus. The Twitter corpus is the result of web scraping and contains language-specific classifications. However, this corpus has the drawback of including inappropriate or offensive words commonly found on gambling websites and free forums. In our testing, the token dictionary size used approximately 30,000 sub-tokens for both the sentence piece and byte-level byte-pair encoding (BPE) methods.

3.2.4. Test results and discussion of self attention on the IM-BERT model

After training the model for approximately 30% of the agreed-upon steps, it exhibited a promising performance. When tested on the same dataset used in the previous model, there was an improvement of around 2%-4% in the exact match (EM) and F1 metrics (the harmonic mean of precision and recall). Thus, at the 30% trained model stage, the model achieved an EM performance of 84.6% and F1 performance of 86.2%. These results indicate that the model surpassed the performances of the RoBERTa and ALBERT models, which served as our baseline during the initial progress. It is worth noting that these performance metrics continued to increase as the model was trained further until it reached 100% completion.

The string-change operation was utilized to calculate the number of modifications required to assess the similarity between strings. The number of differences was determined by summing up the changes made during each operation. Three operations were used to modify strings: i) deletion operation: removing characters from a word to align it with the target string (T) in relation to the source string (S); ii) insertion operation: adding a character at a specific index to align the source string (S) with the target string (T); and iii) exchange operation, that is, replacing a character to align the source string (S) with the target string (T). The testing process for the string-change operation was used to evaluate the performance of the implemented research. These performance values can serve as references for future research and system development. In our system testing, we employed EM and F1-score test metrics. The EM test metric measures the ratio of correctly calculated answers that matches the answers in the data. In contrast, the F1-score test metric combines precision and recall. Furthermore, we applied K-Fold cross-validation to determine the best EM and F1-Score values from various data combinations. The data-sharing ratio used was 80% for the training data and 20% for the development data.

Based on the data presented in Table 2, the performance of the proposed IM-BERT model exhibited relatively good results compared to the baseline model. During the training process, RoBERTa outperformed the proposed IM-BERT model. This was expected because RoBERTa was specifically designed to handle Indonesian datasets. However, during testing, IM-BERT showed superior performance compared with RoBERTa, albeit with a small difference in values.

One factor contributing to the lower performance of the proposed language model was the occurrence of a drastic drop in the loss value, referred to as the “Trough” phenomenon. This occurred because some data had similar structures, making it easier for the model to make accurate predictions. The fluctuating loss value did not indicate a decrease in the capacity of the neural network. This was primarily due to loading the corpus data sequentially instead of randomly, which was necessitated by the limited resources. Consequently, the corpus had to be loaded into separate parts rather than all at once. Certain improvements must be made to the proposed BERT model. First, the string-matching implementation process needs enhancement to enable more effective self-attention, which simplifies the significant transformation processes. In addition, there is a need to incorporate knowledge related to COVID-19. At the time of writing this document, there were limited freely accessible Indonesian language data on COVID-19, in contrast to the availability of scientific research data on COVID-19 in English. Translating knowledge directly from English to Indonesian without proper curation by linguists could lead to misunderstandings and misperceptions.

Table 2. Proposed comparison table of IM-BERT with baseline

Method	Training		Testing	
	EM	F1	EM	F1
IM-BERT	84.6%	86.2%	85.7%	87.6%
ABERTA	84.1%	85.9%	85.1%	86.9%
RoBERTa	84.7%	86.3%	85.6%	87.5%

4. CONCLUSION

The combination of the Rabin-Karp algorithm with the Jaro-Winkler distance algorithm demonstrated higher accuracy than the combination of the Damerau-Levenshtein distance algorithm and the Jaro-Winkler distance algorithm. The use of k-grams in the Rabin-Karp algorithm and the complementary similarity processes of these two algorithms contributed to the generation of appropriate words or sentences.

Furthermore, the fuzzy string-matching model with the modified word feature structure successfully utilized a combination of the Rabin-Karp algorithm and the Jaro-Winkler distance algorithm. This was evident in the formation of the self-attention mechanism, which is the IM-BERT language model capable of analyzing words with various singular and plural sentence patterns. The performance of IM-BERT was evaluated using metrics such as EM, with a score of 85.7% and F1-score of 87.6%, surpassing the baseline model by 0.1%.

ACKNOWLEDGEMENTS

The author expresses gratitude to the Research Center for Data and Information Sciences, National Research and Innovation Agency (BRIN), for providing facilities during the manuscript development process.

REFERENCES

[1] A. Seongyong, H. Hong, H. Kim, J.-H. Ahn, D. Baek, and S. Kang, "A hardware-efficent multi-character string matching architecture using brute-force algorithm," in *2009 International SoC Design Conference (ISOCC)*, 2009, pp. 464–467, doi: 10.1109/SOCDC.2009.5423922.

[2] A. M. A. Ibrahim and M. E. Mustafa, "Comparison criteria between matching algorithms texts application on (horspool's and brute force algorithms)," *Journal of Advanced Computer Science and Technology*, vol. 4, no. 1, Apr. 2015, doi: 10.14419/jacst.v4i1.4283.

[3] P. Manikandan and D. Ramyachitra, "PATSIM: Prediction and analysis of protein sequences using hybrid Knuth-Morris Pratt (KMP) and Boyer-Moore (BM) algorithm," *Gene*, vol. 657, pp. 50–59, May 2018, doi: 10.1016/j.gene.2018.02.069.

[4] D. Shapira and A. Daptardar, "Adapting the Knuth-Morris-Pratt algorithm for pattern matching in Huffman encoded texts," *Information Processing & Management*, vol. 42, no. 2, pp. 429–439, Mar. 2006, doi: 10.1016/j.ipm.2005.02.003.

[5] Z. Xiong, "A composite Boyer-Moore algorithm for the string matching problem," in *2010 International Conference on Parallel and Distributed Computing, Applications and Technologies*, Dec. 2010, pp. 492–496, doi: 10.1109/PDCAT.2010.58.

[6] R. Rahim, A. S. Ahmar, A. P. Ardyanti, and D. Nofriansyah, "Visual approach of searching process using Boyer-Moore algorithm," *Journal of Physics: Conference Series*, vol. 930, Dec. 2017, doi: 10.1088/1742-6596/930/1/012001.

[7] O. Sychev, "Open-answer question with regular expression templates and string completion hinting," *Software Impacts*, vol. 17, Sep. 2023, doi: 10.1016/j.simpa.2023.100539.

[8] D. Belazzougui and M. Raffinot, "Approximate regular expression matching with multi-strings," *Journal of Discrete Algorithms*, vol. 18, pp. 14–21, Jan. 2013, doi: 10.1016/j.jda.2012.07.008.

[9] S. ShabnamHasan, F. Ahmed, and R. Surovi Khan, "Approximate string matching algorithms: a brief survey and comparison," *International Journal of Computer Applications*, vol. 120, no. 8, pp. 26–31, Jun. 2015, doi: 10.5120/21247-4048.

[10] R. Rodriguez-Torrealba, E. Garcia-Lopez, and A. Garcia-Cabot, "End-to-end generation of multiple-choice questions using text-to-text transfer transformer models," *Expert Systems with Applications*, vol. 208, Dec. 2022, doi: 10.1016/j.eswa.2022.118258.

[11] C. Zhai and S. Wibowo, "A systematic review on cross-culture, humor and empathy dimensions in conversational chatbots: the case of second language acquisition," *Heliyon*, vol. 8, no. 12, Dec. 2022, doi: 10.1016/j.heliyon.2022.e12056.

[12] N. A. I. Omoregbe, I. O. Ndaman, S. Misra, O. O. Abayomi-Alli, and R. Damaševičius, "Text messaging-based medical diagnosis using natural language processing and fuzzy logic," *Journal of Healthcare Engineering*, pp. 1–14, Sep. 2020, doi: 10.1155/2020/8839524.

[13] X. Wang, H. Zhang, and Z. Xu, "Public sentiments analysis based on fuzzy logic for text," *International Journal of Software Engineering and Knowledge Engineering*, vol. 26, pp. 1341–1360, Nov. 2016, doi: 10.1142/S0218194016400076.

[14] J. P. Carvalho, F. Batista, and L. Coheur, "A critical survey on the use of fuzzy sets in speech and natural language processing," in *2012 IEEE International Conference on Fuzzy Systems*, Jun. 2012, pp. 1–8, doi: 10.1109/FUZZ-IEEE.2012.6250803.

[15] L. Yujian and L. Bo, "A normalized levenshtein distance metric," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, no. 6, pp. 1091–1095, Jun. 2007, doi: 10.1109/TPAMI.2007.1078.

[16] J. Faes, J. Gillis, and S. Gillis, "Phonemic accuracy development in children with cochlear implants up to five years of age by using Levenshtein distance," *Journal of Communication Disorders*, vol. 59, pp. 40–58, Jan. 2016, doi: 10.1016/j.jcomdis.2015.09.004.

[17] Y. Chaabi and F. A. Allah, "Amazigh spell checker using Damerau-Levenshtein algorithm and N-gram," *Journal of King Saud University-Computer and Information Sciences*, vol. 34, no. 8, pp. 6116–6124, Sep. 2022, doi: 10.1016/j.jksuci.2021.07.015.

[18] C. Zhao and S. Sahni, "String correction using the Damerau-Levenshtein distance," *BMC Bioinformatics*, vol. 20, Jun. 2019, doi: 10.1186/s12859-019-2819-0.

[19] T.-T.-Q. Tran, T.-C. Phan, A. Laurent, and L. DrOrazio, "Improving hamming distance-based fuzzy join in MapReduce using bloom filters," in *2018 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, Jul. 2018, pp. 1–7, doi: 10.1109/FUZZ-IEEE.2018.8491658.

[20] S. D. Galbraith and L. Zobernig, "Obfuscated fuzzy hamming distance and conjunctions from subset product problems," in *Theory of Cryptography*, Springer International Publishing, 2019, pp. 81–110.

[21] M. Pota, M. Esposito, and G. De Pietro, "Learning to rank answers to closed-domain questions by using fuzzy logic," in *2017 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, Jul. 2017, pp. 1–6, doi: 10.1109/FUZZ-IEEE.2017.8015745.

[22] D. Castells-Rufas, "GPU acceleration of Levenshtein distance computation between long strings," *Parallel Computing*, vol. 116, Jul. 2023, doi: 10.1016/j.parco.2023.103019.

[23] M. Li, G. Wang, S. Liu, and J. Yu, "Multi-keyword fuzzy search over encrypted cloud storage data," *Procedia Computer Science*, vol. 187, pp. 365–370, 2021, doi: 10.1016/j.procs.2021.04.075.

[24] J. E. J. Aben, A. C. Timmermans, F. Dingyloudi, M. M. Lara, and J.-W. Strijbos, "What influences students' peer-feedback uptake? Relations between error tolerance, feedback tolerance, writing self-efficacy, perceived language skills and peer-feedback processing," *Learning and Individual Differences*, vol. 97, Jul. 2022, doi: 10.1016/j.lindif.2022.102175.

[25] S. McKeown and W. J. Buchanan, "Hamming distributions of popular perceptual hashing techniques," *Forensic Science International: Digital Investigation*, vol. 44, Mar. 2023, doi: 10.1016/j.fsidi.2023.301509.

[26] R.-D. Li, H.-T. Ma, Z.-Y. Wang, Q. Guo, and J.-G. Liu, “Entity perception of two-step-matching framework for public opinions,” *Journal of Safety Science and Resilience*, vol. 1, no. 1, pp. 36–43, Sep. 2020, doi: 10.1016/j.jnlssr.2020.06.005.
[27] D. Chang, M. Ghosh, S. K. Sanadhya, M. Singh, and D. R. White, “FbHash: A new similarity hashing scheme for digital forensics,” *Digital Investigation*, vol. 29, pp. S113–S123, Jul. 2019, doi: 10.1016/j.diin.2019.04.006.
[28] A. Bouguettaya *et al.*, Eds., *Web information systems engineering-WISE 2017*, vol. 10569, Cham: Springer International Publishing, 2017.
[29] S. C. Cahyono, “Comparison of document similarity measurements in scientific writing using Jaro-Winkler distance method and paragraph vector method,” *IOP Conference Series: Materials Science and Engineering*, vol. 662, no. 5, Nov. 2019, doi: 10.1088/1757-899X/662/5/052016.
[30] I. Ahamed, M. Jahan, Z. Tasnim, T. Karim, S. M. S. Reza, and D. A. Hossain, “Spell corrector for Bangla language using Norvig’s algorithm and Jaro-Winkler distance,” *Bulletin of Electrical Engineering and Informatics (BEEI)*, vol. 10, no. 4, pp. 1997–2005, Aug. 2021, doi: 10.11591/eei.v10i4.2410.
[31] A. P. U. Siahaan *et al.*, “Combination of levenshtein distance and rabin-karp to improve the accuracy of document equivalence level,” *International Journal of Engineering and Technology*, vol. 7, no. 2.27, Jun. 2018, doi: 10.14419/ijet.v7i2.27.12084.
[32] A. B. Khoir, H. Qodim, B. Busro, and A. R. Atmadja, “Implementation of rabin-karp algorithm to determine the similarity of synoptic gospels,” *Journal of Physics: Conference Series*, vol. 1175, Mar. 2019, doi: 10.1088/1742-6596/1175/1/012120.

BIOGRAPHIES OF AUTHORS

Wiwin Suwarningsih graduated from her bachelor’s degree at informatics program, Adityawarman Institute of Technology Bandung in 1996. She got her master’s degree in 2000 at the informatics study program and doctoral degree in 2017 at the School of Electrical and Informatics Engineering, Bandung Institute of Technology. Currently, she works at the Research Center for Data and Information Sciences, National Agency of Research and Innovation (BRIN, Indonesia). Her research interests are artificial intelligence, computational linguistics, Indonesian natural language processing and text mining, information retrieval and question answering systems. Since 2017, she has a project about the Indonesian medical information retrieval using deep learning to support of high-quality medical information. One of the study’s objectives is to provide early detection system for chili varieties. She can be contacted at email: wiwin.suwarningsih@brin.go.id.

Nuryani graduated from her bachelor’s degree at Computer Science, Gadjah Mada University (UGM) in 2003. She got her master’s degree at electrical engineering, Bandung Institute of Technology (ITB) in 2013. Currently, she is a doctoral candidate of the School of Electrical and Informatics Engineering, Bandung Institute of Technology and works at the National Agency of Research and Innovation (BRIN, Indonesia). Her research interests are computer network, deep learning, and natural language processing especially in sentiment analysis. She can be contacted at email: nury012@brin.go.id.